

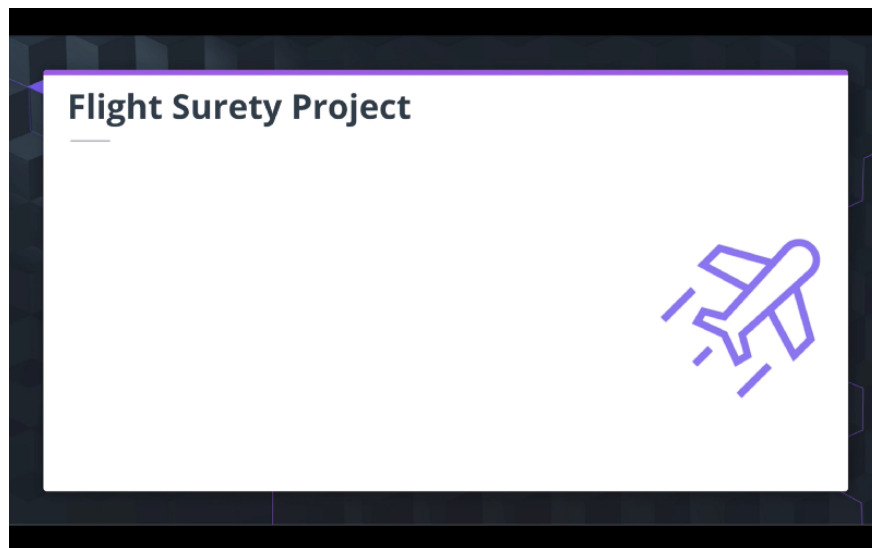
Flight Surety Project FAQ—Udacity Blockchain



Alvaro Andres Pinzon Cortes

Follow

Mar 13 · 4 min read



1. **For the flightsurety project can a passenger purchase insurance for more than one flight?**

Yes

2. **I get the following error while testing**

```
truffle(develop)> test ./test/flightSurety.js Using network 'develop'.
```

```
Error: FlightSuretyApp contract constructor expected 1 arguments, received 0 at
C:\Users\avant\AppData\Roaming\npm\node_modules\truffle\build\webpack:\packages\truffle-contract\contract.js:390:1 at
new Promise (<anonymous>) at
C:\Users\avant\AppData\Roaming\npm\node_modules\truffle\build\webpack:\packages\truffle-contract\contract.js:374:1 at
process._tickCallback (internal/process/next_tick.js:68:7)
truffle(develop)> ''
```

Yes you need to pass the address of the Data contract to FlightSuretyApp in *testConfig.js*

```
let flightSuretyData = await FlightSuretyData.new();  
let flightSuretyApp = await  
FlightSuretyApp.new(flightSuretyData.address);
```

3. I am using openzeppelin 2.2.0 that uses solidity 0.5.2, but the latest version of truffle (5.0.8) uses solidity 0.5.0, how to solve this issue?

These are my suggestions

1) In the truffle configuration file put the version of the compiler that you need to use, in this case 0.5.2

<https://truffleframework.com/docs/truffle/reference/configuration#compiler-configuration>

2) Use a version of open-zeppelin that uses solidity 0.5.0. In here you can find all of the versions

<https://github.com/OpenZeppelin/openzeppelin-solidity/releases>

When you find it install it like this: `npm i openzeppelin-solidity@<version>`

4. I am getting an error . It says sender does not have enough funds to send tx.


One option is that you are using an address that is not generated using the mnemonic that you are using with Ganache, therefore that address is never funded. The solution is then to the mnemonic to match the address or to change the address to match the mnemonic.

Second option. You are using an account generated using the mnemonic, but the account that you are using was not funded by ganache, therefore you need to indicate ganache to fund more accounts to include that account. If you are using ganache-cli use this command to add 40 funded accounts:

```
ganache-cli -p 7545 -m "candy maple cake sugar pudding cream
honey rich smooth crumble sweet treat" -a 40
```

- `-p` or `--port` : Port number to listen on. Defaults to 8545.
- `-m` or `--mnemonic` : Use a bip39 mnemonic phrase for generating a PRNG seed, which is in turn used for hierarchical deterministic (HD) account generation.
- `-a` or `--accounts` : Specify the number of accounts to generate at startup

If you are using the ganache with GUI do this to add 40 funded accounts:



The screenshot shows the Ganache application window. The top navigation bar includes tabs for ACCOUNTS, BLOCKS, TRANSACTIONS, and LOGS. The ACCOUNTS tab is active, displaying a table of accounts. The table has columns for ADDRESS, BALANCE, TX COUNT, and INDEX. The first account is highlighted. The settings icon in the top right corner is circled in red.

ADDRESS	BALANCE	TX COUNT	INDEX
0xc867a8c519D6ed15C682e6A2B03b2c369095d8F0	100.00 ETH	7	0
0x67f2E20Ea36458F311EA5d534e9C0D3d9453e778	100.00 ETH	0	1
0x8235465BfbcBAad4d3a6fC9f2887EA971E1163fB	100.00 ETH	0	2
0xC4b2D12C7e2C4903cb6fbcAd51619d3994b457e6	100.00 ETH	0	3
0xB3DDd89c1680ac70a8fd20C16d17eeaE70d65b59	100.00 ETH	0	4
0x2361022A309703dcaA4F24Cfc318EbA8799a5467	100.00 ETH	0	5
0xEC7c85505301b4F59415143e47494F31dd9160Aa	100.00 ETH	0	6

SERVER

HOSTNAME
127.0.0.1 - localhost

PORT NUMBER
8545

NETWORK ID
5777

AUTOMINE
☐

ERROR ON TRANSACTION FAILURE
☐

ACCOUNTS & KEYS

ACCOUNT DEFAULT BALANCE
100

TOTAL ACCOUNTS TO GENERATE
40

AUTOGENERATE HD MNEMONIC
☒

report space gravity rule tourist hungry elevator nose spatial hamster exerci

LOCK ACCOUNTS
☐

5. Are there any specific version requirements for this project?

The problems works well with this versions

```
Truffle v5.0.8 (core: 5.0.8) Solidity - ^0.4.24 (solc-js)
Node v10.7.0 Web3.js v1.0.0-beta.37
```

6. When I run each test file in a separate command it works:

```
truffle test ./test/flightSurety.js
truffle test ./test/oracles.js
```

But when I run the test files in a single command I get an error:

```
truffle test
```

Error:

***Error: the tx doesn't have the correct nonce. account has nonce of: 2
tx has nonce of: 3***

There is apparently a problem when running the 2 test files at the same time: `truffle test`. This causes a problem with the nonces. It works fine and it is **ACCEPTABLE** to run each test file separately: `truffle test ./test/flightSurety.js` `truffle test ./test/oracles.js`

The **IMPORTANT** thing is that the test files pass all the test cases if they are run in separate commands like this:

```
truffle test ./test/flightSurety.js
truffle test ./test/oracles.js
```

7. It is not clear if for an airline needs to be funded to be able to register another airline. According to the instructions it looks like it just need to be registered but that could leave the system into a strange state. Is it correct?

It is expected that an airline is FUNDED to be able to register another airline

8. Trying to deploy FlightSuretyData, but geeting an error: “FlightSuretyData”—Invalid number of parameters for “undefined”. Got 0 expected 1!

If the constructor of a contract needs parameters, then you should manage this in the migration files. This is an example of how to pass as parameter to the FlightSuretyApp contract the address of the FlightSuretyData contract:

2_deploy_contracts.js

```
const FlightSuretyApp =
  artifacts.require("FlightSuretyApp");
const FlightSuretyData =
  artifacts.require("FlightSuretyData");
const fs = require('fs');
module.exports = function(deployer) {
  let firstAirline =
    '0xf17f52151EbEF6C7334FAD080c5704D77216b732';

  deployer.deploy(FlightSuretyData)
    .then(() => {
      return deployer.deploy(FlightSuretyApp,
        FlightSuretyData.address)
    })
    .then(() => {
      //OTHER CODE FROM THE BOILERPLATE
      //OTHER CODE FROM THE BOILERPLATE
    })
}
```

Boilerplate

udacity/FlightSurety

Udacity Blockchain Nanodegree Course 6 "Flight
Surety" Project - udacity/FlightSurety
github.com



9. In my code, when I try to refund a passenger, I do it with a function

```
function withdraw() external requireFunds { uint256 value =
  flightSuretyData.getFunds(msg.sender);
  flightSuretyData.setFunds(msg.sender, 0); // TODO:
  msg.sender.transfer(value); emit FundsWithdrawn(msg.sender,
  value); }
```

With ``msg.sender.transfer(value)``. But always fails. Why?

The `withdraw` function is normally implemented in the `flightSuretyData` contract

Because `flightSuretyData` is supposed to store the ether

Maybe that is the problem, `flightSuretyData` has the funds and `flightSuretyApp` don't, then `flightSuretyApp` don't not have enough funds to send to the passenger

Also remember that according to the architecture of `contract upgradability`, the contract that should store the Dapp's data is `flightSuretyData`. It also applies that `flightSuretyData` stores the funds because it is a central resource that is stored and managed and that is supposed to live as long as the system

